1

WRDC-TR-90-8007
Volume VIII
Part 14

AD-A248 922

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 14 - Virtual Terminal Unit Test Plan

L. Jones, F. Glandorf

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

[illegible]
DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

Bruce Rasmussen
BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

Unclassified

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UTP620344300	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 14
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City, State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 78011F; PROJECT NO. 595600; TASK NO. F95600; WORK UNIT NO 20950607
11. TITLE Virtual T See block 19	

12. PERSONAL AUTHOR(S)
Structural Dynamics Research Corporation: Jones, L., Glandorf, F.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr.,Mo.,Day)*	15. PAGE COUNT
Final Report	4/1/87-12/31/90	1990 September 30	21

16. SUPPLEMENTARY NOTATION
WRDC,MTI Project Priority 6203

17. COSATI CODES

FIELD	GROUP	SUB GR.
1308	0905	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This unit test plan establishes the methodology and procedures used to adequately test the capabilities of the Virtual Terminal Protocol and the computer programs implementing it.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VIII -User Interface Subsystem

Part 14 - Virtual Terminal Unit Test Plan

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371 22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

UTP620344300
30 September 1990

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

GENERAL

1.1 Purpose

This unit test plan establishes the methodology and procedures used to adequately test the capabilities of the Virtual Terminal protocol and the computer programs implementing it (the Virtual Terminal software). The Virtual Terminal is one configuration item of the Integrated Information Support System User Interface.

1.2 Reference Documents

[1] American National Standards Institute, Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986, 26 March 1986.

[2] American National Standards Institute, Code Extension Techniques for Use with the 7-Bit Coded Character Set of American National Standard Code for Information Interchange, ANSI X3.41-1974, 14 May 1974.

[3] American National Standards Institute, Additional Controls for Use with American National Standard Code for Information Interchange, ANSI X3.64-1979, 18 July 1979.

[4] American National Standards Institute, Hollerith Punched Card Code, ANSI X3.26-1980, 2 May 1980.

[5] International Organization for Standardization, Information Processing - ISO 7-Bit Character Set for Information Interchange, ISO 646-1983.

[6] International Organization for Standardization, ISO 7-Bit and 8-Bit Coded Character Sets - Code Extension Techniques, ISO 2022-1982.

[7] International Organization for Standardization, ISO 7-Bit and 8-Bit Coded Character Sets - Additional Control Functions for Character - Imaging Devices, ISO 6429-1983.

[8] ICAM Documentation Standards, IDS150120000C, 15 September 1983.

[9] Structural Dynamics Research Corporation, IISS Terminal Operator Guide, OM 620144000, 1 November 1985.

[10] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620144200B, 1 November 1985.

[11] Structural Dynamics Research Corporation, IISS Form Processor User Manual, UM 620144200B, 1 November 1985.

[12] Structural Dynamics Research Corporation, Virtual Terminal Development Specification, DS 570144300, 6 June 1988.

[13] Structural Dynamics Research Corporation, Virtual Terminal User Manual, UM 620144300B, 1 November 1985.

[14] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620144700, 1 November 1985.

[15] Boeing Military Airplane Company, NTM Programmer's Manual, PRM620242000, 20 July 1987.

[16] Structural Dynamics Research Corporation, Form Processor Unit Test Plan, UTP620344200, 15 December 1987.

[17] Structural Dynamics Research Corporation, Virtual Terminal Product Specification, PS 570144300A, 6 June 1988.

1.3 Terms and Abbreviations

AI: Application Interface.

American Standard Code for Information Interchange: the character set defined by ANSI X3.4 and used by most computer vendors.

AP: Application Process.

Application Interface: subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The Application Interface enables applications to be hosted on computers other than the host of the User Interface.

Application Process: a cohesive unit of software that can be initiated as a unit to perform some function or functions.

ASCII: American Standard Code for Information Interchange.

Attention Interrupt: an asynchronous signal usually generated by pressing a special key or combination of keys which indicates that the currently executing program should be stopped.

Attribute: visual and logical characteristics (e.g. blinking, highlighted, enterable).

Block Mode: a mode of interaction in which a device does not send characters as they are entered but waits for some triggering event to send an entire block of characters.

Character Set: the characters used by a computer system and the bit patterns assigned to represent them.

Class Routines: routines which are common to a number of systems or devices.

Computer Program Configuration Item: an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Control Character: a control function which is coded as a single character.

Control Function: an action that affects the recording, processing, transmission, or interpretation of data.

Control Sequence: a sequence of characters beginning with the Control Sequence Introducer control function which represents a control function.

Control String: a sequence of characters beginning and ending with a delimiting control function which represents a control function.

CPCI: Computer Program Configuration Item.

DD: Device Driver.

Device Driver: software which handles input and output for a specific kind of terminal by mapping between terminal specific commands and Virtual Terminal commands.

Device Routines: routines which are specific to a particular type of terminal.

EBCDIC: Extended Binary Coded Decimal Interchange Code.

Escape Sequence: a sequence of characters beginning with the Escape control character which represent a control function.

Extended Binary Coded Decimal Interchange Code: the character set used by a few computer vendors (notably IBM) instead of ASCII.

Field: two dimensional space on a terminal screen which contains data.

Fill Area: a polygonal area the interior of which is filled-in in some fashion.

Host: a computer system running the Integrated Information Support System.

IISS: Integrated Information Support System.

Integrated Information Support System: a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. IISS addresses the problems of integration of data resident on heterogeneous databases supported by heterogeneous computers interconnected via a local area network.

Line: a group of points which are connected in the order specified.

Marker: a group of points the locations of which are indicated by a symbol.

Master Mode: the mode of operation used by a Device Driver which is started by a user using host operating system commands in order to connect into IISS.

Network Transaction Manager: IISS subsystem that performs the coordination, communication, and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

NTM: Network Transaction Manager.

Numeric Parameter: a parameter which represents a numeric value.

Operating System: software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Physical Device: a hardware terminal.

Primitive: a graphic element contained in a view. Either a fill area, line, marker, or text.

Script File: a file containing Virtual Terminal commands which can be replayed to duplicate the session during which it was recorded.

Selective Parameter: a parameter which is used to represent selections from a list of possible values.

Slave Mode: the mode of operation used by a Device Driver which is started by the User Interface.

Software Control String: a control string which is intended to be used to control software as opposed to hardware.

System Routines: routines which are specific to a particular computer system or operating system.

Text: a primitive consisting sequence of characters the size, color, orientation, and location of which may be specified.

UI: User Interface.

UIM: User Interface Monitor.

UIMS: User Interface Management System.

User Interface: IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System and the User Interface Management System.

View: an area of the screen which can contain primitives.

Virtual Terminal: subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface: the callable interface to the Virtual Terminal.

VT: Virtual Terminal.

VTI: Virtual Terminal Interface.

Window: an area of the screen which an contain other windows (subwindows), fields, and views.

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor.

SECTION 2

DEVELOPMENT ACTIVITY

2.1 Statement of Pretest Activity

During system development, the computer programs were tested progressively. Functionality was incrementally tested and, as bugs were discovered by this testing, the software was corrected.

Each Virtual Terminal function was tested individually during Virtual Terminal development.

All pretesting activity was conducted by the individual program developer in a manual mode. The developer manually entered data onto the screen and observed the results. Any errors were noted by the developer, and corrections to the Virtual Terminal software were then made after a testing session.

2.2 Pretest Activity Results

The pretest activity was very successful in the elimination of programming bugs.

SECTION 3

SYSTEM DESCRIPTION

3.1 System Description

The Virtual Terminal protocol and software allow Integrated Information Support System applications to run on a wide variety of physical devices (terminals). The Virtual Terminal protocol defines the available functions and attributes and the Virtual Terminal software translates between the Virtual Terminal protocol and commands for a particular type of terminal. An implementation of the Virtual Terminal software for a specific operating system and terminal type is referred to as a Device Driver.

3.1.1 Interface Requirements

Application programs normally interface with the User Interface rather than the Virtual Terminal, but a set of Application Interface routines are provided for special situations where a direct interface is required. An application program using this direct interface must call the Virtual Terminal initialization (INITVT) and termination (TERMVT) routines to bypass the Form Processor. Since INITVT and TERMVT merely toggle Form Processor bypass mode, INITFP and TERMFP must be called at the beginning and end of the application.

An application that uses the Virtual Terminal normally calls the routines INITVT, GETVTI, PUTVTI, and TERMFP of the Application Interface, but since routines with identical calling sequences are part of the Form Processor callable routines, testing may be simplified by linking directly with these routines instead. Figures 3-1 and 3-2 illustrate the normal Virtual Terminal interfaces and the interfaces for testing purposes.

3.2 Testing Schedule

As figure 3-2 indicates, the execution of the Virtual Terminal is dependent on a small part of the Form Processor subsystem (the callable routines INITVT, GETVTI, PUTVTI, and TERMVT) and its interface to the Virtual Terminal. Since the Form Processor subsystem is dependent on the whole Virtual Terminal subsystem, testing of the Virtual Terminal subsystem should occur prior to testing of the Form Processor subsystem.

```
            +------------------------+
            |  Application Program   |
            +------------------------+
            | Application Interface  |
            +------------------------+
                        &
                        | UI Request / Response
                        |
            +------------------------+
            |     User Interface     |
            +------------------------+
                        &
                        | VT Command / Response
                        |
            +------------------------+      )
            |       VT Monitor       |      |
            +------------------------+      |
            |     Class Routines     |      |
            +------------------------+      }  Device Driver
            |    Device Routines     |      |
            +------------------------+      |
            |    System Routines     |      |
            +------------------------+      )
                        &
                        | Device Command
                        |
                  +------------+
                  | Terminal   |
                  +------------+
```

Figure 3-1 Normal Virtual Terminal Interfaces

```
          +------------------------+
          | Application Program    |
          +------------------------+
          |   Form Processor       |
          +------------------------+     )
          |   Class Routines       |     |
          +------------------------+     |
          |   Device Routines      |     }  Device Driver
          +------------------------+     |
          |   System Routines      |     |
          +------------------------+     )
                     &
                     | Device Command
                     |
                +-----------+
                | Terminal  |
                +-----------+
```

Figure 3-2 Virtual Terminal Interfaces for Unit Testing

3.3 First Location Testing

These tests of the Virtual Terminal require the following:

Equipment: IISS host computer system, terminals supported on the host by the Virtual Terminal as listed in the Terminal Operator's Guide [9].

Support Software: The Form Processor subsystem of the User Interface.

Personnel: One integrator familiar with IISS.

Training: The Virtual Terminal User manual [13].

Deliverables: The Virtual Terminal Subsystem.

Test Materials: This test may be run interactively by writing and executing a test program with the appropriate Virtual Terminal calls, entering the appropriate data, and observing the output as outlined in this test plan. Alternatively, the Form Processor test application ARTEST may be used to exercise the Virtual Terminal calls without writing a test program. Another alternative is to consider the Form Processor Unit Test specified in [16] to be an adequate test of the Virtual Terminal.

Security considerations: None.

3.4 Subsequent Location Testing

The requirements as listed above need to be met.

SECTION 4

SPECIFICATIONS AND EVALUATIONS

4.1 Test Specification

The following functions should be performed in order to test each Device Driver:

1) Go into Form Processor Bypass Mode.

2) Test the Screen Formatting Functions:
   Record Separator
   Unit Separator
   Set Transmit State
   Set Window Precedence
   Remove Window
   Select Window
   Erase Window
   Define Window
   Define Viewport
   Define Field
   Define View Text
   Define View Fill Area
   Define View Markers
   Define View Lines

3) Test the Input Functions:
   Application Program Command
   Cursor Position Report
   Select Window
   Define Field

4) Test the Cursor Control Functions:
   Backspace
   Horizontal Tab / Cursor Horizontal Tab
   Line Feed / Index / Cursor Down / Vertical Position Relative
   Carriage Return
   Next Line / Cursor Next Line
   Reverse Index / Cursor Up
   Cursor End of Line
   Cursor Forward / Horizontal Position Relative
   Cursor Backward
   Cursor Previous Line
   Cursor Position / Horizontal and Vertical Position
   Cursor Back Tab
   Horizontal Position Absolute
   Vertical Position Absolute

5) Test the Data Entry and Modification Functions:
   Form Feed / Next Page / Previous Page
   Graphic Characters
   Insert Character
   Erase Display
   Erase Line

Insert Line
Delete Line
Erase Field
Delete Character
Erase Character
Set Mode
Reset Mode

6) Test the Miscellaneous Functions:
Bell
Refresh Screen
Reset to Initial State
Media Copy

7) Exit Form Processor Bypass Mode.

Figure 4-1 maps these functional requirements to test activities.

4.2 Testing Methods and Constraints

The tests as outlined in Section 5 must be followed. The required calling sequence is stated for the test program. This testing uses the normal mode of operation of these functions and does not completely exercise all the error combinations that a user of the Virtual Terminal might create by faulty entry of command sequences. Much of this testing has occurred, however, during the normal testing done by the developers of these functions. No data recording is required. No additional constraints are placed on this unit test besides those listed in Section 3.3 of this unit test plan.

	Test Activities									
Functional Requirements	A	B	C	D	E	F	G	H	I	J
Enter Form Processor Bypass Mode	X									
Screen Formatting Functions		X		X		X		X		
Input Functions			X		X		X			
Cursor Control Functions			X		X		X			X
Data Entry/Modification Functions			X		X		X			X
Miscellaneous Functions										X
Exit Form Processor Bypass Mode									X	

Figure 4-1 Functional Requirements Mapping

Key to Figure 4-1

A - Call to INITVT
B - First call to PUTVTI
C - First call to GETVTI
D - Second call to PUTVTI
E - Second call to GETVTI
F - Third call to PUTVTI
G - Third call to GETVTI
H - Fourth call to PUTVTI
I - Call to TERMVT
J - Keyboard input

4.3 Test Progression

The progression of testing of the Virtual Terminal is fully outlined in Section 5 of this unit test plan. This progression should be followed exactly to ensure the successful testing of this IISS configuration item.

4.4 Test Evaluation

The test results are evaluated by comparing the screens to that specified as successful for the given test, as outlined in Section 5.

SECTION 5

TEST PROCEDURES

5.1 Test Description

This test is performed by writing a test program which uses the callable interface to exercise the functionality of the Virtual Terminal. In section 5.3 the required function calls and the required parameters are documented for each function being tested and the resulting successful results are also documented.

5.2 Test Control

The order of the testing is also documented in section 5-3. The test control information is completely described by the sequence of calls.

5.3 Test Procedures

I) The program which should be written to test the Virtual Terminal should contain the following calls:

```
CALL "INITFP".

CALL "INITVT" USING RCODE.

CALL "INQLDV" USING LDV-ID, RCODE.
  In the following, XX must be replaced with the returned
value of LDV-ID
  converted to a character string.

CALL "PUTVTI" USING BUFFER, LENGTH, RCODE.

 Where BUFFER contains
  SW  - Select Window:    <ESC>[XXs
  EW  - Erase Window:     <ESC>[u
  DW  - Define Window:
<ESC>[900;1;1;80;23;0;0;80;23;0;41;37w
  SW  - Select Window:    <ESC>[900s
  DV  - Define Viewport:  <ESC>[32767;0;65535;32767v
  DF  - Define Field:     <ESC>[12;10;20;5;0;0;7;33;44x
  DF  - Define Field:     <ESC>[10;15;10;1;1;0;1x-Testing1-
  DVT - View Text:
<ESC>[2000;60000;2;0;100;100;5000;0;0;0;0;0;10
                          "v-Testing2-
  DVF - View Fill Area:
<ESC>[5;1;2000;50000;20000;30000;20000;50000;
                          2000;30000&v
  DVM - View Markers:
<ESC>[6;2;100;32000;40000;42000;40000*v
  DVM - View Markers:
<ESC>[4;3;200;32000;35000;42000;35000*v
  DVL - View Lines:
<ESC>[7;1;100;2000;30000;20000;10000;20000;
```

```
                          30000;2000;10000;2000;30000-v
  DVL - View Lines:
<ESC>[0;3;200;32000;30000;50000;10000;50000;
                          30000;32000;10000;32000;30000-v
  RS  - Record Separator: <RS>
 and LENGTH is 434 plus twice the length of XX

CALL "GETVTI" USING BUFFER, MAX_LENGTH, LENGTH, RCODE.
 Where MAX_LENGTH = Length of BUFFER

CALL "PUTVTI" USING BUFFER, LENGTH, RCODE.
 Where BUFFER contains
  SW - Select Window:    <ESC>[XXs
  DW - Define Window:    <ESC>[902;1;1;80;23;0;0;80;23;0w
  SW - Select Window:    <ESC>[902s
  DF - Define Field:     <ESC>[2;10;20;15;0;0;7x
  DF - Define Field:     <ESC>[20;15;10;1;1;0;5x-Testing3-
  RS - Record Separator: <RS>
 and LENGTH is 86 plus the length of XX

CALL "GETVTI" USING BUFFER, MAX_LENGTH, LENGTH, RCODE.
 Where MAX_LENGTH = Length of BUFFER

CALL "PUTVTI" USING BUFFER, LENGTH, RCODE.
 Where BUFFER contains
  SW - Select Window:         <ESC>[XXs
  WP - Set Window Precedence: <ESC>[900p
  US - Unit Separator:        <US>
  RS - Record Separator:      <RS>
 and LENGTH is 11 plus the length of XX

CALL "GETVTI" USING BUFFER, MAX_LENGTH, LENGTH, RCODE.
 Where MAX_LENGTH = Length of BUFFER

CALL "PUTVTI" USING BUFFER, LENGTH, RCODE.
 Where BUFFER contains
  SW - Select Window:    <ESC>[XXs
  RW - Remove Window     <ESC>[900r
  RS - Record Separator: <RS>
 and LENGTH is 10 plus the length of XX

CALL "TERMVT" USING RCODE.

CALL "TERMFP".
```

II) Link this program with Form Processor and Virtual Terminal libraries (see Appendix A).

III) Run your program

a) The first screen should have a red background and contain the following:
   - An input field beginning at row 12 column 10 which is 20 columns wide and 5 rows deep. It should be yellow (reverse video on monochrome terminals.
   - An output field beginning at row 10 column 15 which is 10 columns wide and 1 row deep. It should contain the

string "-Testing1-" and be bold.
- A green graphics text string "-Testing2-".
- A magenta "bow tie" which is filled in.
- Two cyan plus signs.
- Two blue stars which are twice as big as the plus signs.
- A white "bow tie" which is not filled in.
- A dotted black "bow tie" which is not filled in.

b) Move the cursor down, then to the right, then up to the top left corner of the screen.

c) Depress Tab.

d) The cursor should move to the input field; enter something. The entered characters should be yellow.

e) Depress a function key. The buffer received by GETVTI should contain a series of Set Window commands, a Define Field command, and the data which was entered. This should be followed by an Application Program Command and a Cursor Position Report command (see [13]).

f) The second screen should have two fields on it:
- An input field beginning at row 2 column 10 which is 20 columns wide and 15 rows deep. It should be reverse video.
- An output field beginning at row 20 column 15 which is 10 columns wide and 1 row deep which is initialized to "-testing3-". It should be blinking.

g) Without entering anything, depress a function key. The buffer received by GETVTI should contain a series of Set Window commands followed by Define Field commands. There should be no data after the Define Field commands. The last data in the buffer should be an Application Program Command and a Cursor Position Report command.

h) The next screen should be the same as the previous one.

i) At this time you should test out the remaining cursor control functions, data entry and modification functions, and miscellaneous functions.

j) Depress a function key to terminate the application.

Appendix A

Sample link Command Descriptions

LINK FOR TEST PROGRAM

```
$ if p1 .eqs. "" then p1 = "VT100"
$ if p2 .eqs. "" then p2 = "nomap"
$ link/exe=[]test test/'p2',-
  iiss_prod:[xxxx.ui.safp]safp/lib,-
  [xxxx.ui.fp]fp/lib,-
  [xxxx.ui.test]stubs/lib,-
  [xxxx.ui.driver]driver/lib/inc='p1',-
  [xxxx.ui.clib]clib/lib
```

NOTE: Replace 'xxxx' with the IISS test directory.